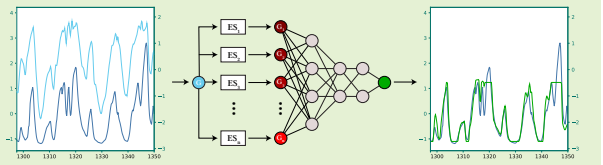
Machine Learning Based Compensation for Inconsistencies in Knitted Force Sensors

Roland Aigner and Andreas Stöckl

***Abstract*— Knitted sensors frequently suffer from inconsistencies due to innate effects such as offset, relaxation, and drift. These properties, in combination, make it challenging to reliably map from sensor data to physical actuation. In this paper, we demonstrate a method for counteracting this by applying processing using a minimal artificial neural network (ANN) in combination with straightforward pre-processing. We apply a number of exponential smoothing filters on a re-sampled sensor signal, to produce features that preserve different levels of historical sensor data and, in combination, represent an adequate state of previous sensor actuation. By training a three-layer ANN with a total of 8 neurons, we manage to significantly improve the mapping between sensor reading and actuation force. Our findings also show that our technique translates to sensors of reasonably different composition in terms of material and structure, and it can furthermore be applied to related physical features such as strain.**



***Index Terms*— filtering, force sensor, knitting, machine learning, neuronal networks, resistive sensing, textile sensor**

I. Introduction

TEXTILE based sensors are of high interest in research and industry due to numerous beneficial properties, such as lightness, breathability, and potential stretchability. In particular, knits are inherently elastic textiles, due to their geometric composition of courses of interlocking loops, as opposed to weaves, where yarn is travelling straight. This elasticity makes them ideal for sensing stress or strain [1]–[4] that generally perform according to Holm's theory [5], which states that contact resistance R depends on material resistivity ρ, material hardness H, contact point count n, and pressure P, with

$$R = \frac{\rho}{2}\sqrt{\frac{\pi H}{nP}}\,.$$

Consequently, the overall sensor resistance drops when pressure at the loop intermeshing points' contacts is increased, e.g., by straining or pressing. However, depending on the yarn material properties and/or structural composition of a knitted structure, knitted sensors usually suffer from considerable inconsistencies that have to be addressed [3], [6]–[8], such as settling effects, offset, overshooting, hysteresis, as well as long- and short-term sensor drift, some of which we speculate are due to slight structural re-arrangements of the yarn within the fabric. Inherently, the raw measurement signal tends to seriously deviate from the desired output, i.e., the applied force. This is undesirable in several use cases since it complicates downstream analysis of raw sensor data. Unfortunately, since these effects are unlike common noise, traditional methods such as frequency-domain- or Kalman-filters are insufficient to get rid of those.

Figure 1 (top) shows an example timeline plot of a recording of applied force and resulting sensor reading. Long-term drift is most eminent, however short-term inconsistencies are also apparent when zooming in (cf. Figure 1 bottom left, for a magnification of three timeline snippets), which illustrate that a basic mapping, e.g., by multi-point calibration is impractical and not promising. Due to the nature of a knitted fabric, effects like latency, hysteresis, drift, offset, overshooting, etc. (and the interaction thereof) are innate. Furthermore, as their extent depends also on the chosen knitting structure, which makes their use challenging for scenarios where reliability is required.

Our main hypothesis for this work is that inconsistencies that are reflected in the sampling data are in fact ultimately deterministic, as their cause is in the knits' physical and geometrical composition, however too complex to analyze or model manually from empirical observations. Hence, our approach is to utilize an artificial neural network (ANN) to learn and model those factors instead. A further objective is to keep the computational complexity low by reducing the number of required features and by using small-scale networks, so our method is viable for low-end embedded devices with highly limited computational capabilities.

In this paper, we present an easy-to-implement method for mitigating this error, which is based on a small-scale multilayer perceptron neural network (MLP NN) and comes with little modeling/training effort and low computational cost. NNs are in general frequently used for modeling complex non-

Submitted for Review June 15, 2023

Roland Aigner is with the Interactive Media department of the Upper Austria University of Applied Sciences, 4232 Hagenberg, Austria. (e-mail: roland.aigner@fh-hagenberg.at).

Andreas Stöckl is head of the Interactive Media department of the Upper Austria University of Applied Sciences, 4232 Hagenberg, Austria. (e-mail: andreas.stoeckl@fh-hagenberg.at).

This article has been accepted for publication in IEEE Sensors Journal. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/JSEN.2023.3346686

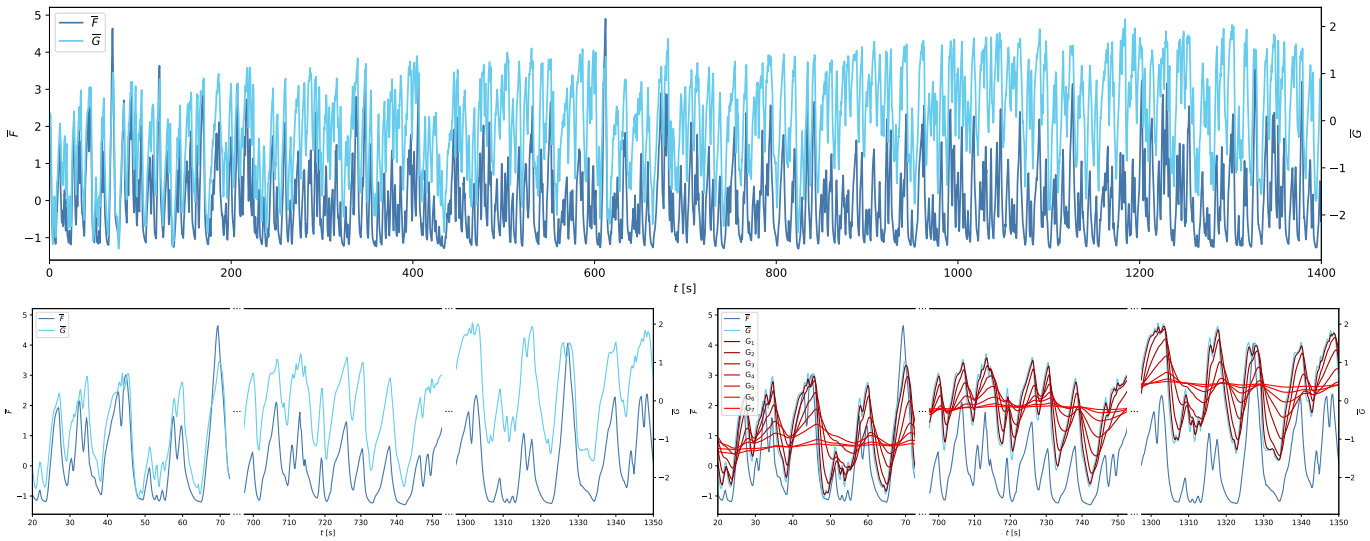


Fig. 1: Timeline of one of our recordings using the PES sensor (top): Overlaying normalized trends of ground truth $\overline{F}$ and sensor readings $\overline{G}$ clearly shows an upwards drift of the sensor readings. Zooming into three segments of the timeline (bottom left), reveals inconsistencies and even contradicting effects, such as overshooting vs. underestimation in the first segment. Our method utilizes a neural network that is trained using several smoothed sensor signals as features (bottom right). We combine a set of those with different smoothing factors to incorporate different degrees of historic information.

linear systems, as it has been shown that even small networks are able to well approximate any continuous function of n real variables [9]. Moreover, MLP networks are relatively easily trained using Backpropagation (BP) algorithms [10].

The main contributions of this paper are as follows:

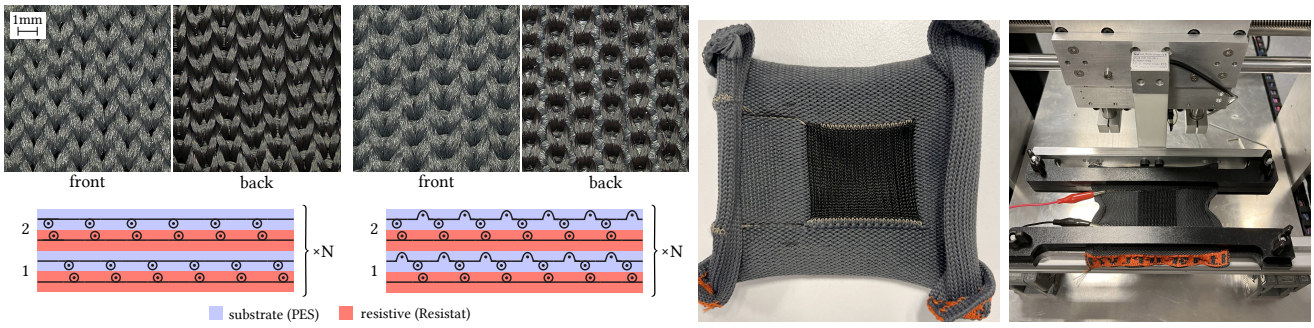
- a method for rectifying inherent inconsistencies in raw sensor data that result from the structural nature of knitted sensors.
- description of a related feature-set that we used as an input vector for an ANN that model several levels of historic sensor data.
- mitigation of short-term errors, as well as removal of long-term sensor drift, that go beyond sensor hysteresis.
- an exemplary processing pipeline including a NN of minimal complexity that is easily trained and computationally undemanding during operation.
- a demonstration of the method's transferability to sensor knits of different structure and to different objectives (e.g., mapping sensor data to stain instead of stress).

While there are numerous works that used Machine Learning techniques for classification, e.g., for detecting hand gestures [11], sitting postures [12], or exercise activities [13], others have used neural networks in scenarios similar to ours, however mostly for compensating hysteretic behavior. Dang et al. [14] used Radial Basis Function (RBF) NNs to model Preisach hysteresis of piezoceramic actuators. Similarly, Lien et al. [15] used hysteretic recurrent NNs in the context of piezoelectric actuators, modeling hysteresis in the neurons' activation function. Tong et al. [16] used Backlash-Based Hysteresis Simulation Models to test a NN that approximates hysteretic non-linearities. Wu et al. [17] implemented a dynamic NN structure based on the Hammerstein model for dynamic error compensation of infrared thermometer sensors. More recently, Weiss et al. [18] applied Kalman filters for preprocessing chemical sensors' data for downstream machine learning. Jondhale et al. [19] combined Kalman filters with General Regression NNs for 2D-position tracking from RSSI signals. In the field of textile-based sensing, Atitallah et al. [20] compared filtering methods such as moving average, moving median, Savitzky-Golay, and Gaussian for processing data from a sensor glove that incorporated CNT-based sensors. Vu et al. [21] implemented an adaptive fuzzy-NN for capacitive pressure sensors, which were based on spacer-knit structures. Finally, Liu et al. [22] utilized RBF NNs to compensate for hysteresis disturbance in non-affine, nonlinear systems, however the test set consists of limited, generated data that is furthermore repetitive.

While all those works are related, objective, material, and use cases differ from ours and hence the techniques are hardly transferable. To our knowledge, there is so far no related work applying NNs for data rectification of knitted piezoresistive stress/strain sensors targeted at random, real-world actuation.

II. Sensor Implementation

Our sensors utilize a widespread knitting pattern that is often, however inconsistently, called "Twill" in the textile industry, due to its structural similarity with a Twill Weave. It consists of courses with alternating knit and float stitches, shifted by one needle every other row (cf. Figure 2a) . The high number of floats results in exceptional stability along course-direction and high elasticity along wale-direction, when compared to more straightforward knits, such as Plain or Double Jersey [23]. In this regard it exhibits a characteristic similar to a Cardigan, however with orthogonal anisotropic behavior, which can be an advantage in certain use case scenario, where omni-directional elasticity is not desired.



(a) (b) (c)

Fig. 2: Closeups and knitting patterns of the Twill based knitting structures (a): for the PES- (tubular structure, left), and Lycra (connected, i.e., PES tucked to back-face Resistat, right). PES sensor patch (b), with conductive yarn traces connecting the resistive area (black) on both upper and lower ends. We evaluated our sensors using a custom-built tensile tester which is equipped with a force cell (c).

Since a Twill can be knitted on a single needle bed, the opposite bed is available for additional structures. We decided to apply our knitted sensors on a substrate carrier structure, by knitting a Twill consisting of PES on the front bed and another Twill using resistive yarn on the back bed, to have the sensor part completely covered by the PES on one side for protecting it from abrasion. Furthermore, this gave us better control in balancing for a uniform stability across the whole fabric, when compared to integrating a sensor patch as an Intarsia [11]. Note that this requires the two faces to be connected to not fall apart; we did this by tucking the resistive yarn to the PES at the outer wales of the sensor area produces a tubular structure (cf. Figure 2a, left).

From a previous study [24], we learned that adding Lycra to the substrate can significantly improve elastic recoil, minimizing hysteresis. We therefore fabricated two variations: one with pure PES-substrate, one with additional Lycra. Since the resistive face is not complemented with Lycra, we tightly connected front and back faces for these sensor patches, by tucking the substrate to the sensor knit for every loop (cf. Figure 2a, right), to prevented interference from resistive face's lagging behind the elastic substrate. We decided to include this additional version in our evaluation, to estimate how well our method translates to sensors of different design.

For the substrate carrier, we used a den 150 PES from TWD Fibres GmbH, and for the Lycra thread we used a 140 Lycra core covered with PES den 150/20 from Jörg Lederer GmbH. For the resistive sensing areas, we used Polyester-based, Carbon-sheathed Resistat P6204[1] from Shakespeare® with den 100/24 and ∼10 MΩ/m. For the connector traces, we used silver-coated PA-yarn Madeira HC40[2], with den 260 and <300 Ω/m. All our patches were knitted on a flat-bed knitting machine of type ADF 530-32 KI W Multi Gauge from KARL MAYER STOLL, at gauge E 7.2. For more details regarding materials and fabrication of our sensors, we refer to [24].

[1]https://shakespeare-pf.com/product/polyester/
[2]https://www.shieldex.de/products/madeira-hc-40/

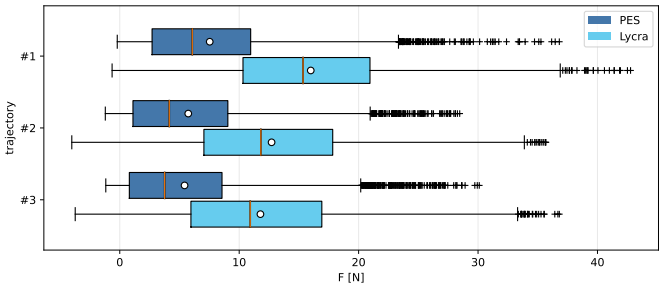


Fig. 3: Due to different knitting structures and material compositions, the two sensors operate in different force ranges when actuated using identical trajectories.

III. Data Acquisition

For collecting training and test data, we used a custom-made tensile tester, which we built from a CNC milling machine (cf. Figure 2c). The clamps for attaching the patches at both ends featured needles at 2 cm distance to secure the textile against slipping. The moving actuator was equipped with a single-point load-cell with nominal load of 10 kg (Sauter CP 10-3P1[3]) and was sampled at ∼40 Hz by an ADS 1231 24-bit Delta-Sigma ADC[4]. Since sensor resistance readings were slightly noisier, we supersampled with 128 Hz via a simple voltage divider with a 606 kΩ reference resistor. We used an Adafruit ADS1115 16-bit ADC[5] for sampling, buffered readings and averaged values in windows of ∼25 ms, again resulting in a rate of ∼40 Hz for our final samples. Measurements for force and resistance, as well as actuator displacement and timestamps were captured into CSV files by the MCU firmware. A single ESP32 on an Adafruit HUZZAH32 Feather board[6] was used for sampling and recording to SD card.

The tensile tester was controlled by Art-Soft Mach4 CNC Control Software (v4.2.0), running on a Windows 10 PC. To simulate pseudo-natural motion for reasonably representative

[3]https://www.kern-sohn.com/shop/en/products/measuring-technology-components/CP-10-3P1/
[4]https://www.ti.com/product/ADS1231
[5]https://www.adafruit.com/product/1085
[6]https://www.adafruit.com/product/3405

TABLE I: Statistics of our three generated actuator trajectories. Velocities were around 1 mm/s, with a maximum of 4.930 mm/s.

	v [mm/s]			a [mm/s²]		
trajectory	max	mean	SD	max	mean	SD
#1	4.142	0.943	0.593	19.05	1.397	1.303
#2	4.930	0.979	0.617	18.61	1.497	1.343
#3	4.039	1.030	0.644	19.25	1.546	1.414

actuation of the sensors and collecting corresponding sensor data, we generated G-code trajectories based on Perlin Noise [25], [26], to control the actuator along the x-axis. Our objective was to generate non-repetitive trajectories, so we could guarantee our resulting model would not merely learn to repeat a specific pattern and instead be applicable for random actuation. We created three of those trajectories, to get one data set for training our ML models and two for testing their performance. Based on pre-evaluation of the sensors [24], we chose the amplitudes of our trajectories to move in an approximate range of 0% to 30% extension. Maximum, mean, and SD of velocities and accelerations can be found in Table I. We collected data for both of our sensors, i.e., PES-only ("PES") and Lycra-enhanced ("Lycra") variants. Note that this resulted in different force ranges, due to the difference in firmness (cf. Figures 3, bottom). Our objective was to test our method on sensors of slightly different structure and material, to get an estimation of its portability between sensor designs. The total duration of our actuation process and the time-span we recorded was ∼23 minutes each.

IV. Data Processing

For reasons of simplicity, our presented method was not designed to take timing into account yet, hence it requires uniform sampling periods. However, we noticed the sampling frequency was not perfectly constant (μ=41.5 Hz, σ=14.2), as a consequence of our multi-component setup. We further noticed the targeted frame rate of ∼40 Hz was higher than what our method required, since results did not significant change after downsampling to 20 Hz. Resampling with even lower rates (e.g., 10 Hz) seemed to yield worse results, though, we therefore re-sampled our recorded force and resistance data to 20 Hz using linear interpolation in between the sample points.

The main objective of our work is to infer force-data from the raw measurements that were taken from the sensor, meaning both trends should be as identical as possible, which is not the case with raw measurement data (cf. Figure 1). Hence, we argue the coefficient of determination r^2 is a reasonable metric for quantifying this property and hence to judge about the performance of our approach[7]. However, since force F and sensor resistance R are inversely proportional, we utilize the sensor conductivity $G = 1/R$ instead of the resistance. Furthermore, since both F and G cover largely different ranges, we normalized both to identical ranges to $\overline{F}$ and $\overline{G}$, by removing mean and scaling to unit variance using the *StandardScaler* from the *scikit learn* Python package[8]. This pre-processing step is also beneficial (and in fact recommended [27]) for better performance of machine learning estimators later on. Hence, we calculate the coefficient of determination with

$$r^2(X,Y) = 1 - \frac{\sum_i (x_i - y_i)^2}{\sum_i (x_i - \mathtt{mean}(Y))^2}$$

substituting X with normalized force measurement $\overline{F}$, and Y with pre-processed and normalized conductivity measurement $\overline{G}$. To quantify the performance of our machine learning model, we then use the prediction p for Y instead.

After pre-processing our raw data, we calculated our initial r^2 scores as baseline values with 0.423, 0.471, and 0.526 for the recordings with our PES version and as 0.703, 0.667, and 0.734 for those with the Lycra version (cf. Table IV). It is already eminent that the performance of our Lycra patch is superior, which is in line with previous findings [24]. However, we first focus on the most basic implementation, without additional Lycra, which we expect to benefit most from improvement by computational means.

As mentioned, our hypothesis is that an ANN is able to model the knit's state and infer the actuation from an (seemingly randomly) inconsistent sensor reading. For example, it seems reasonable that sensor offset and settling speed is affected by recent elongation. From that follows, that there needs to be historic data available for the current prediction; this can either be implemented by a feedback mechanism, or otherwise by choosing input features that include temporal information. For sake of simplicity, we decided for the latter. Using a number of n previous samples within a certain time-frame as features is not promising since this would result in a very high number of features and would therefore rapidly increase complexity of computation and network topology, increasing the risk of overfitting. Furthermore, this approach would highly depend on the sample-rate. Instead, we decided for providing historic data in the form of several smoothed signals, with varying degrees of responsiveness. We utilized exponential smoothing [28] with

$$y(t) = \alpha x_t + (1-\alpha) y(t-1)\,,$$

where α is a smoothing factor in range [0 1]; i.e., the lower the value for α, the higher the drag. We noticed initialization with $y(0) = x_0$ introduced too much bias for the signals with high drag, therefore we initialized with the mean of samples values within a window of M samples, $y(0) = \mathtt{mean}(x_0 \dots x_{M-1})$. For calculating an adequate window-size that is depending on drag and sample rate f, we empirically found $M = \lceil 1/f\alpha \rceil$ delivers reasonable initialization values. By filtering the sensor conductivity $\overline{G}$ with a set of smoothing factors α_i, we gain a set of N filtered sensor signals G_i, reflecting different degrees of temporal data (cf. Figure 1, bottom right), which represent the elements of our feature vector. Note that our exponential smoothing implementation does not take timing into account,

[7]Note that we use the less-common lower-case notation r^2 to avoid confusion with the sensors' electrical resistance R.

[8]https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html

TABLE II: Smoothing base value a and factor count N were varied for determining factors α_i, e.g., a = 10 and N = 3 result in A = (10^{-1}, 10^{-2}, 10^{-3}). The values for our initial factors that we used as baseline were adjusted manually and therefore not calculated from a and N.

a	N	$A = (\alpha_1 \dots \alpha_N)$	
-	-	(0.5, 0.1, 0.025, 0.0025)	baseline
2.5	4	(0.4, 0.16, 0.064, 0.0256)	$\alpha_i = 1/a^i$
2.5	7	(0.4, 0.16, …0.0016384)	
2.5	10	(0.4, 0.16, …0.000104858)	
5	4	(0.2, 0.04, 0.008, 0.0016)	
5	7	(0.2, 0.04, …0.0000128)	
10	3	(0.1, 0.01, 0.001)	
10	4	(0.1, 0.01, 0.001, 0.0001)	

TABLE III: Number of hidden layers' (HL) neurons were determined by building the products of *base sizes* with *topology* vectors and flooring the results, e.g., $\lfloor 6 \times (½, 1, ¼) \rfloor$ would give three layers with 3, 6, and 1 neurons, respectively. After removing all identical permutations as well as those containing sizes <2, 114 unique variations remained.

parameter	variations
HL base sizes	2, 3, 4, 6, 8, 12, 16, 32
topologies	(1, 1), (1, ½), (1, ¼), (1, 1, 1), (1, 1, ½), (1, ½ , ½), (1, ½ , ¼), (½ , 1, 1), (½ , 1, ½), (½ , 1, ¼), (1, 1, 1, 1), (1, 1, 1, ½), (1, 1, ½ , ½), (1, ½ , ½ , ½), (1, ½ , ½ , ¼), (1, ½ , ¼ , ¼), (1, ¼ , ¼ , ¼), (½ , 1, 1, 1), (½ , 1, 1, ½), (½ , 1, ½ , ½), (1, ½ , ½ , ½)

which is a further reason we re-sampled our data to a constant rate of 20 Hz.

Multi-layer Perceptrons (MLP) trained by back-propagation (BP) algorithms are commonly used for function approximation, we therefore used the *MLPRegressor*[9] of *scikit learn*, with *relu* activation function and maximum iterations of 10,000. Note that MLPs are particularly sensitive to feature scaling [29], which makes our previously described pre-processing mandatory.

We started with experimental α values of 0.5, 0.1, 0.025, and 0.0025, which gained promising results, therefore we kept this set as a baseline. From there, we tried different sets of smoothing factor vectors $A = (\alpha_1 \dots \alpha_N)$, with $\alpha_i = 1/a^i$, modifying a and N to get several sets of different sizes and granularity. Since we found that data smoothed with α values below 10^{-4} held too little information, we mostly refrained from going beyond those. Apart from feature vectors, we varied the ANN's hidden layer *sizes* (i.e., neuron counts) and *topologies*. We did not go beyond neuron counts of 32 and beyond 4 hidden layers, since we started to notice frequent overfitting at these values. We then empirically tested all permutations[10] of those parameters to find the optimal configuration (cf. Table III for a complete listing).

[9]https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPRegressor.html

[10]after removing duplicates and those including layers with less than 2 neurons, both due to flooring the products, 114 network permutations remained. Testing those with all 8 variations of A, this resulted in 912 candidates for our pipeline.

TABLE IV: r^2 of our initial (pre-processed) and predicted data. We present results of our two test sets (A, B), as well as the training sets (t), which are non-representative, but are included for sake of completeness. PES show highest gain from our approach; the Lycra patches ultimately yield higher scores.

	r²(F,G)	r²(F,p)	gain
PES_t	0.423	(0.781)	(85%)
PES_A	0.471	0.791	68%
PES_B	0.526	0.767	46%
$Lycra_t$	0.703	(0.841)	(20%)
$Lycra_A$	0.667	0.830	24%
$Lycra_B$	0.734	0.828	13%

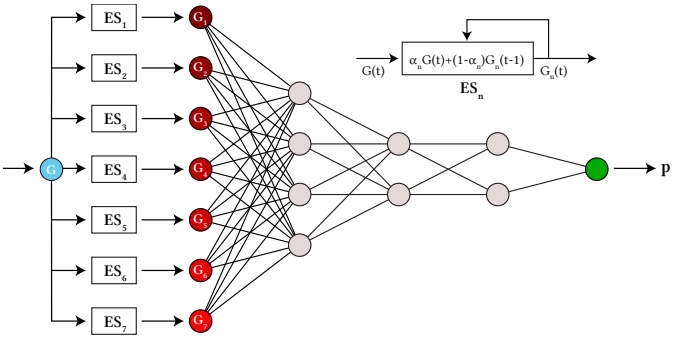


Fig. 4: We feed pre-processed (re-sampled and normalized) data $\overline{G}$ into a number of exponential smoothing filters using different smoothing factors and use the results as feature vector for our neural network. In our tests, the combination of a = 2.5 and N = 7 with hidden layer sizes (4, 2, 2) resulted in the best r^2 score.

V. Evaluation and Discussion

A. Results

Our tests showed that a low number of hidden layers worked well in most cases and increasing them did not considerably improve scores. Occasionally, adding a fourth hidden layer did even degrade results, but this seems also to be subject to network topology. Overall, hidden layer *base sizes* of at least 4 were required, otherwise the resulting models would be unusable. In terms of features, we saw that smoothing factors resulting from a = 10 (with both N = 3 and N = 4) did not perform well. We speculate this is because the smoothing factors are too far apart and the resulting low number of G_i features include too little historical information. Overall, our variations with a = 5 performed better, however the best scores were achieved with a = 2.5 and N <10. A spreadsheet including all the scores of our network variations can be found in the supplementary material.

Our systematic tests resulted in the best r^2 score for the parameter-combinations of a = 2.5, N = 7 (i.e., A = (0.4, 0.16, 0.064, 0.0256, 0.01024, 0.004096, 0.0016384), HL base size = 4, and topology = (1, ½, ½), which gives neuron counts of (4, 2, 2) (cf. Figure 4). Predictions of force values from our training sets resulted in r^2 values of 0.791 and 0.767 (PES), as well as 0.830 and 0.828 (Lycra), producing highest gains for the PES variants. Figure 5 shows the result of test set #1, with p overlaid on $\overline{G}$ and $\overline{F}$. It is striking that the long-term drift

was removed. Inspection of section snippets (cf. Figure 5, left) reveal considerable improvement over the pre-processed input signal that goes well beyond what could be achieved with more basic transformation techniques. Using identical parameters, the network was also trained for the Lycra test-set, and as can be seen in Figure 5 (right), there is similar improvement. In general, however, we can observe occasional underestimation of peaks, while the model seems to perform very well at low-force areas.

B. Further Experiments

A reasonable question at this point is whether or not our method translates to different kinds of input data. Since the majority of related work utilizes knitted sensors not primarily as *force* sensors but rather as *strain* sensors, we ran our recordings of actuator offset through the same pipeline to see if our technique also translates to this objective. As mentioned above, d also drifts over time which may be the initial cause of this long-term drifting effect, since G does not considerably drift relative to d, however both trends still differ significantly, with initial r^2-scores between 0.260 and 0.319 (PES), as well as 0.337 and 0.490 (Lycra). Using our unmodified processing pipeline as presented in Figure 4, we were able to boost those scores up to 0.699 (PES), and 0.669 (Lycra) for the test sets (further details can be found in the supplement). This implies that our method works exceptionally not only for force as a main metric.

A related question concerns variation of actuation speed and amplitudes, since for our main study, we generated trajectories that feature similar statistic values for (cf. Table I), in order to assess consistency of our results. To estimate how well the proposed method translates beyond that specific data, we generated more trajectories with strain values up to 50%, actuator velocities up to 14 mm/s (mean: 2.87 mm/s, SD: 2.06 mm/s), and accelerations up to 65.54 mm/s^2 (mean: 9.27 mm/s^2, SD: 7.87 mm/s^2) and recorded data using the PES patches. We noticed that initial r^2 scores were already higher (median: 0.644), suggesting that sensors are more consistent for higher actuation speeds. With applying our model, we could still boost the r^2 by 15% (median). One extreme case was a rise from 0.349 to 0.795, thus a gain of 128%. However, we judged that G_1, i.e., the NN input feature with least smoothing, already lagged too much from the original. We countered this by reducing smoothing, lowering a from 2.5 to 1.75, which resulted in $\alpha_1 = 0.57$, etc., which resulted in a better r^2 gain of 24% (median). Summarizing, this shows that our presented method does translate to different data, however, fine-tuning smoothing factors against an estimated range can still be beneficial. In yet another experiment, we applied a NN trained with data from the initial trajectories to the data with the newly generated ones. Using the initial values for smoothing factors, we were able to increase r^2 by 23% (median), meaning already pre-trained networks can also be reasonably applied across different data sets exhibiting different statistic distributions.

In terms of feature choice would like to note that we tried several variations, e.g., including additional information: first, we experimented with slope values, i.e., first derivatives of each of the smoothed signals G_i, as well as a set of smoothed first derivatives of G, however we found those did not improve prediction quality significantly as they seem redundant. Second, we briefly experimented with features taken from the frequency domain by calculating windowed FFTs, however the signal turned out to contain little information beyond very low frequencies and we did not go into great lengths to exploring this direction any further. In terms of alternative machine learning methods, we experimented with linear[11], polynomial[12] (3$^{\text{rd}}$ and 4$^{\text{th}}$ order), and random forest regressors[13], but were not able to produce results of comparable quality.

Furthermore, we briefly tried re-sampling to rates other than 20 Hz, namely half and twice the frequency. We got slightly worse results with 10 Hz and similar results with 40 Hz, so we kept our initial value of 20 Hz for data pre-processing.

C. Discussion & Limitations

The results show that our approach performs remarkably well in improving the mapping from sensor reading to physical actuation. The fact that our processing pipeline is exceptionally small and requires little computation effort, makes it viable for applications in highly limited environments and platforms. We would like to note that due to the nature of our method that is based on features from smoothed sensor readings, as opposed to resorting to a learned hysteresis-model, it is reasonable to assume that our method adapts well to permanent effects, such as chemically and mechanically induced material degradation.

We showed that our method proves successful also when applied to sensors with considerable differences in structure and elastic behavior. Furthermore, the network can be trained to successfully predict not only force but also strain values. Further experiments with actuation of varying strain amplitudes and speeds suggest that our method translates well between different input data, although fine-tuning parameters can be beneficial. We believe that this can be evaded by increasing the number of features G_i, with smaller smoothing factor steps in-between. This will increase computational complexity and possibly add redundancy, however, the resulting model could be more versatile. However, we believe it may be possible to infer optimal a and N from a given data set with reasonable effort. We see great value in implementing a fully adaptive system this way and plan to investigate in this direction in future work.

We do acknowledge a few limitations of this work. First, we did not go into great length in investigating entirely different ANN topologies, such as Deep Belief Networks [30], Extreme Learning Machines [31], Echo State Networks [32], etc. We do not expect serious performance gains by changing the topology type, however it will be an interesting direction to explore, and we leave this for future work.

[11] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

[12] https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PolynomialFeatures.html

[13] https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html

This article has been accepted for publication in IEEE Sensors Journal. This is the author's version which has not been fully edited and
content may change prior to final publication. Citation information: DOI 10.1109/JSEN.2023.3346686

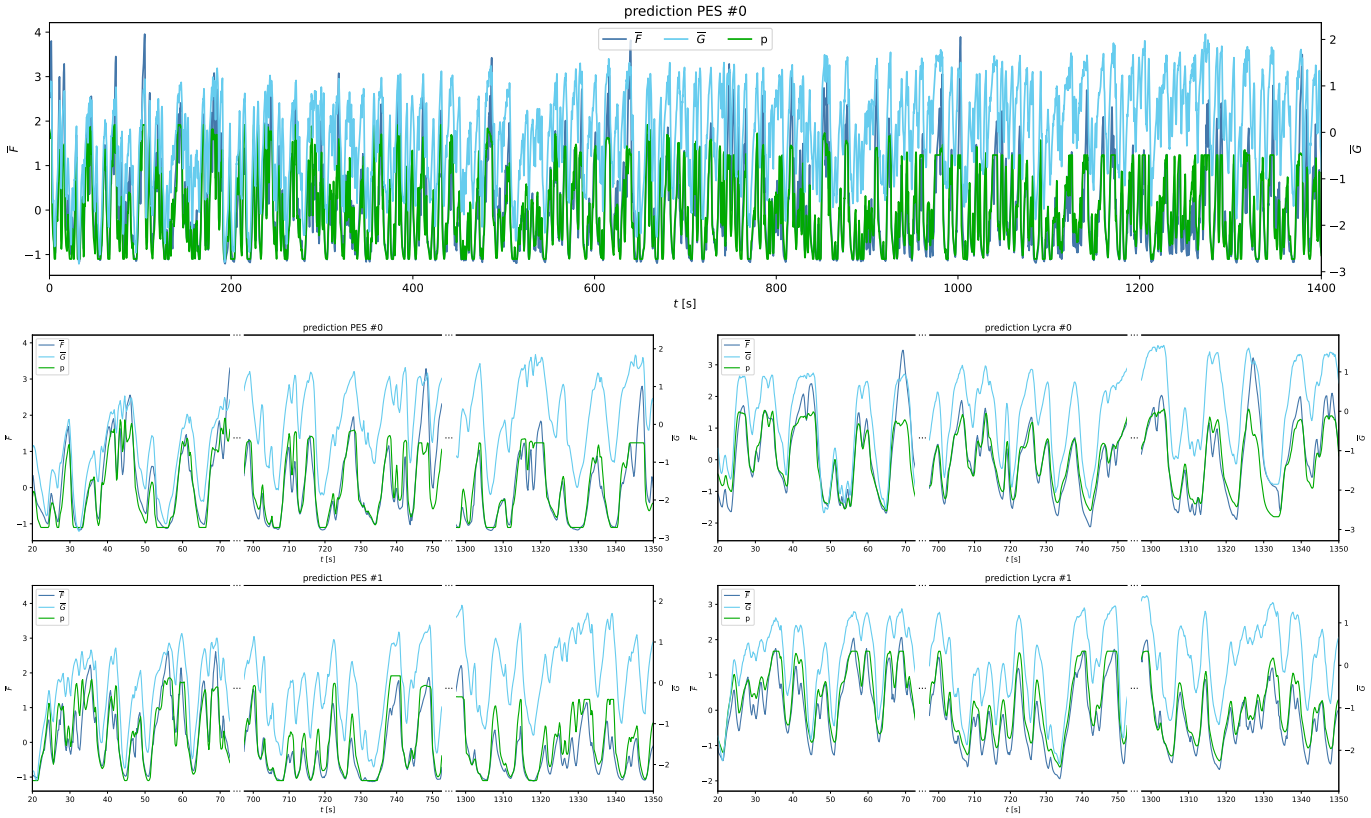


Fig. 5: Our results show that prediction values p rectify sensor inconsistencies G exceptionally well. Most striking is the removal of long-term drift (top). When looking more closely at segments (left), we see that p aligns well with ground truth data F in most cases. The same method also translates well to our Lycra-variants using different yarn material and knitting composition.

Second, we do not have data that goes beyond our roughly 23 minutes recordings. We did observe in prior evaluations, that drift decreases in a logarithmic manner, therefore we expect the most challenging effect of drift is to be addressed at the beginning. Furthermore, our method presents a multi-purpose and adaptive way of handling the issue in that it provides a means of modeling long-term drift in the form of highly smoothed sensor signals (cf. G_6 and G_7 in Figure 1, bottom right) and use this as highly compact and low-complexity features. In terms of signal peaks of high prominence that are sometimes underestimated (cf. Figure 5 PES #0 at second 748) or cropped (PES #1 at second 740), it is reasonable to believe that further training with data that is more specific to this issue will rectify the model accordingly.

Third, we present a method of predicting scaled data. Mapping the range to meaningful physical values will require some initial calibration step. Note that this calibration would also be required without our pipeline, since, e.g., readings in Ohms or Siemens need to be translated to Newtons either way.

Fourth, we mentioned we re-sampled our data to a constant sample-rate, since our exponential smoothing filters are not considering timing data and are therefore sensitive to varying Δt. We believe this can be easily overcome by ensuring a more consistent sample rate in the firmware and by taking timing data into account for smoothing. As mentioned, the actual ADC and firmware would be able to sample at a much higher rate (128 Hz in our case) and the solution comes down to implement a solid down-sampling routine, that outputs at a reasonably constant rate.

Fifth, for this investigation, we initialized our exponential smoothing filters with the mean of the first M samples to avoid biased starting values (cf. Section IV). Applying this in a real-world application would require a short duration for initialization for collecting those samples. However, we did not see this as a major limitation and not as the core of this work. We trust it is not a serious challenge to find alternative initialization methods to set $y(0)$ as there are numerous ways to do so [33], or to substitute the entire smoothing filter, as we do not believe our method relies on this exact method for smoothing.

No doubt, the very best solution to specific sensors may vary in detail, slight modifications of the network and features (e.g., smoothing factors and number of features) may be beneficial for fine-tuning to the scenario at hand. However, we noticed during our experiments that many of those slight adjustments only result in minor improvements that may not be significant or representative and could be subject to the particular training data.

VI. Conclusion

We demonstrated a method of utilizing an ANN for correcting inconsistent sampling data read from a knitted resistive

force sensor, by pre-filtering the raw input signal and thus providing multiple levels of temporal information as a feature vector to the NN. Once trained, the pipeline can be used as a real-time filter for translating sensor readings to physical actuation data. We demonstrated our method using a MLP NN that in its best-performing configuration requires only three hidden layers and a total of 8 neurons to achieve considerable improvement over the input data, which signifies exceptionally low computational requirements and therefore facilitates applications in a wide variety of scenarios. Although we applied the technique to translate raw sensor data to the trend of physical force/stress, the method can be easily translated to related metrics such as strain, as we briefly demonstrate in the supplement. Furthermore, we successfully applied our technique to a knitted sensor of different structure and behavior, which implies it translates well to slightly different conditions, possibly even to entirely different use cases beyond knitted sensors that suffer from similar issues, such as considerable hysteresis and drift.

REFERENCES

[1] E. P. Scilingo, F. Lorussi, A. Mazzoldi, and D. De Rossi, "Strain-sensing fabrics for wearable kinaesthetic-like systems," *IEEE Sensors Journal*, vol. 3, no. 4, pp. 460–467, 2003.

[2] X. Liang, H. Cong, Z. Dong, and G. Jiang, "Size prediction and electrical performance of knitted strain sensors," *Polymers*, vol. 14, no. 12, 2022. [Online]. Available: https://www.mdpi.com/2073-4360/14/12/2354

[3] T. Alam, F. Saidane, A. al Faisal, A. Khan, and G. Hossain, "Smart- textile strain sensor for human joint monitoring," *Sensors and Actuators A: Physical*, vol. 341, p. 113587, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0924424722002254

[4] P.-A. Yang, X. Cui, R. Li, M. Shou, X. Gong, C.-H. Lee, and K. Zhang, "Highly sensitive and selective multidirectional flexible strain sensors with cross-shaped structure based on fe nws/graphene/interlock knit fabric for human activity monitoring," *IEEE Sensors Journal*, vol. 23, no. 19, pp. 23 440–23 447, 2023.

[5] R. Holm, *"Electric Contacts: Theory and Application"*, 4th ed. Heidelberg: Springer Berlin, 1967.

[6] H. Zhang, X. Tao, T. Yu, and S. Wang, "Conductive knitted fabric as large-strain gauge under high temperature," *Sensors and Actuators A: Physical*, vol. 126, no. 1, pp. 129–140, 2006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0924424705005832

[7] A. Grassi, F. Cecchi, M. Maselli, M. Röling, C. Laschi, and M. Cianchetti, "Warp-knitted textile as a strain sensor: Characterization procedure and application in a comfortable wearable goniometer," *IEEE Sensors Journal*, vol. 17, no. 18, pp. 5927–5936, 2017.

[8] B. Bozali, J. J. F. van Dam, L. Plaude, and K. M. B. Jansen, "Development of hysteresis-free and linear knitted strain sensors for smart textile applications," in *2021 IEEE Sensors*, 2021, pp. 1–4.

[9] J. Park and I. W. Sandberg, "Universal Approximation Using Radial-Basis-Function Networks," *Neural Computation*, vol. 3, no. 2, pp. 246–257, 06 1991. [Online]. Available: https://doi.org/10.1162/neco.1991.3.2.246

[10] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Prentice Hall, 2010.

[11] S. Lee, Y. Choi, M. Sung, J. Bae, and Y. Choi, "A knitted sensing glove for human hand postures pattern recognition," *Sensors*, vol. 21, no. 4, 2021. [Online]. Available: https://www.mdpi.com/1424-8220/21/4/1364

[12] Y. Jiang, J. An, F. Liang, G. Zuo, J. Yi, C. Ning, H. Zhang, K. Dong, and Z. L. Wang, "Knitted self-powered sensing textiles for machine learning-assisted sitting posture monitoring and correction," *Nano Research*, vol. 15, no. 9, pp. 8389–8397, Sep 2022. [Online]. Available: https://doi.org/10.1007/s12274-022-4409-0

[13] I. Wicaksono, P. G. Hwang, S. Droubi, F. X. Wu, A. N. Serio, W. Yan, and J. A. Paradiso, "3dknits: Three-dimensional digital knitting of intelligent textile sensor for activity recognition and biomechanical monitoring," in *2022 44th Annual International Conference of the IEEE Engineering in Medicine & Biology Society (EMBC)*, 2022, pp. 2403–2409.

[14] X. Dang and Y. Tan, "An inner product-based dynamic neural network hysteresis model for piezoceramic actuators," *Sensors and Actuators A: Physical*, vol. 121, no. 2, pp. 535–542, 2005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0924424705001937

[15] J. Lien, A. York, T. Fang, and G. D. Buckner, "Modeling piezoelectric actuators with hysteretic recurrent neural networks," *Sensors and Actuators A: Physical*, vol. 163, no. 2, pp. 516–525, 2010. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0924424710003717

[16] Z. Tong, Y. Tan, and X. Zeng, "Modeling hysteresis using hybrid method of continuous transformation and neural networks," *Sensors and Actuators A: Physical*, vol. 119, no. 1, pp. 254–262, 2005. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0924424704007046

[17] D. Wu, S. Huang, W. Zhao, and J. Xin, "Infrared thermometer sensor dynamic error compensation using hammerstein neural network," *Sensors and Actuators A: Physical*, vol. 149, no. 1, pp. 152–158, 2009. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S092442470800530X

[18] M. Weiss, M. S. Wiederoder, R. C. Paffenroth, E. C. Nallon, C. J. Bright, V. P. Schnee, S. McGraw, M. Polcha, and J. R. Uzarski, "Applications of the kalman filter to chemical sensors for downstream machine learning," *IEEE Sensors Journal*, vol. 18, no. 13, pp. 5455–5463, 2018.

[19] S. R. Jondhale and R. S. Deshpande, "Kalman filtering framework-based real time target tracking in wireless sensor networks using generalized regression neural networks," *IEEE Sensors Journal*, vol. 19, no. 1, pp. 224–233, 2019.

[20] B. Ben Atitallah, J. R. Bautista-Quijano, H. Ayari, A. Y. Kallel, D. Bouchaala, N. Derbel, and O. Kanoun, "Comparative study of digital filters for a smart glove functionalized with nanocomposite strain sensor," in *2021 18th International Multi-Conference on Systems, Signals & Devices (SSD)*, 2021, pp. 1366–1371.

[21] C. C. Vu and J. Kim, "Highly elastic capacitive pressure sensor based on smart textiles for full-range human motion monitoring," *Sensors and Actuators A: Physical*, vol. 314, p. 112029, 2020. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S092442471932343X

[22] W. Liu and T. Zhao, "An active disturbance rejection control for hysteresis compensation based on neural networks adaptive control," *ISA Transactions*, vol. 109, pp. 81–88, 2021. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0019057820304146

[23] D. J. Spencer, *Knitting Technology: A Comprehensive Handbook and Practical Guide*, 3rd ed. Woodhead Publishing, 2001.

[24] R. Aigner and F. Hepper, "An evaluation of multi-component weft-knitted twill structures for sensing tensile force," 2023, arXiv:2306.07612.

[25] K. Perlin, "An image synthesizer," in *Proceedings of the 12th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '85. New York, NY, USA: Association for Computing Machinery, 1985, p. 287–296. [Online]. Available: https://doi.org/10.1145/325334.325247

[26] ——, "Improving noise," in *Proceedings of the 29th Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '02. New York, NY, USA: Association for Computing Machinery, 2002, p. 681–682. [Online]. Available: https://doi.org/10.1145/566570.566636

[27] scikit learn, "scikit-learn user guide: 6.3. preprocessing data," 2023, accessed on May 30th, 2023. [Online]. Available: https://scikit-learn.org/stable/modules/preprocessing.html#preprocessing-scaler

[28] E. S. Gardner Jr, "Exponential smoothing: The state of the art," *Journal of forecasting*, vol. 4, no. 1, pp. 1–28, 1985.

[29] scikit learn, "scikit-learn user guide: 1.17. neural network models (supervised)," 2023, accessed on May 30th, 2023. [Online]. Available: https://scikit-learn.org/stable/modules/neural_networks_supervised.html

[30] Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in *Proceedings of the 19th International Conference on Neural Information Processing Systems*, ser. NIPS'06. Cambridge, MA, USA: MIT Press, 2006, p. 153–160.

[31] G.-B. Huang, Q.-Y. Zhu, and C.-K. Siew, "Extreme learning machine: Theory and applications," *Neurocomputing*, vol. 70, no. 1, pp. 489–501, 2006, neural Networks. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0925231206000385

[32] H. Jaeger and H. Haas, "Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication," *Science*, vol. 304, no. 5667, pp. 78–80, 2004. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.1091277

[33] S. Nahmias, *Production and Operations Analysis*, 6th ed. McGraw-Hill/Irwin, 2008.